長野大学紀要　第17巻第4号　16—21頁（322—327頁）1996

ソフトウェア設計における知識獲得と仕様獲得支援

Supporting Knowledge Acquisition and Requirements Acquisition in Software Engineering

谷口道興
Michioki Taniguchi

ソフトウェア工学の誕生以来これまでに、多数の概念や技術的方法が提案されてきた。一方、ソフトウェア開発量の増加および複雑化に伴いより一層の生産性の向上、信頼性の保証が求められているのも事実である。これらの状況にたいして人工知能あるいはその応用分野としての知識工学の成果をソフトウェア開発に応用しようという研究が提案されている。本稿では、ソフトウェア開発方法論に関して知識獲得支援技術の有用性について述べる。

1. 知識獲得・学習技術

ソフトウェアシステムの開発における要求分析・定義の重要性は、システム設計・構築の過程でも指摘されているところであり、これまでも様々な概念・技法・ツールが提案され実用化されてきた[1),2)]。

知識の獲得や学習の過程で、知識の表現の方法を如何にすればユーザのある程度の満足の度合を保証できるのか、同時に工学的、言語学的にも無理の無いシステムが構築可能なのかが問題となる。工学的なシステムとして知識を獲得するには対象領域を明確にした知識に限定することも必要となる。対象領域がある一定の範囲に定められれば、知識の内容は相当に明確になる。知的システムが使う知識の目的は事実の認識や事実解釈の支援、提案による問題解決、原因や理由の同定、矛盾する知識の修正、冗長な知識の整理をすることにある。また、知識の構造化についてはモジュール化の概念があるが、モジュール化された知識は全体として対象領域の構造を前記の目的に合致するよう反映されていなければならない[3)]。

ソフトウェアシステムの開発における要求獲得と同様に、エキスパート・システムの開発においても問題開発に必要な情報を専門家やテキストなどの知識源から収集・定式化し、それを推論機構が処理可能な形式に変換してシステム内に取り込む作業が必要である[4)]。エキスパート・システムにおける知識獲得とは、知識の抽出（収集）、知識の変換、知識の整理・体系化の作業をいう。知識の整理・体系化とは知識相互の関連性を明確化する、知識ベースの不備な部分を補うなどの操作により、知識ベースを質的に整理する作業を意味する。一般に、エキスパート・システム開発の最大のボトルネックは「知識獲得」にある。知識獲得が、エキスパート・システムの構築に不可欠であることを認識すれば、ソフトウェア開発でとくに新しい技術が要求される分野での知識獲得の困難さが浮き彫りとなる。知識獲得は一般にナレッジエンジニヤ（Knowledge Engineer; KE）が対象分野の専門家にインタビューするという形式で進められるが、そもそも人間が自分の知っている知識を誤りなく明確に表現することはできないし、特に、“専門家と呼ばれる人は本人が意識している以上の専門知識を持っている（ファイゲンバウム）”ので、このような意識下にある知識を引き出すことにその本質的な困難さがあり多大な時間を要する作業となっている。同様の問題として、エキスパートシステムに必要な知識ベースには、専門家の知識を誤りなく明確に表現しなければ意味がない。しかしながら、専門家と呼ばれる人の専門知識は経験的な知識を中心に構築され

ているのが通常であり、経験豊富な専門家であればあるほどその知識は無意識的になり、それを明確に表現することもまた困難となる。また、専門家と呼ばれる人は、推論モデルを構築しようとする人に様々な情報を提供してくれるが、その中には、(1)過去の解決した問題に関する経験(2)問題解決のための経験や方法(3)問題解決に用いた方法を選択した理由に関する知識等が含まれる。専門家にとっても過去の問題解決の経験から推論規則を導出することは容易なことではなく、一般に受容されている科学知識と関連付けてそれらの知識を導出することの困難性は増大する。さらに専門家からの知識の獲得があったとしても、それらの知識のコード化も問題となる。

このため、知識獲得の作業を機械化することを目的に、知識獲得支援ツールの開発が行われてきた。知識獲得支援ツールは知識獲得の多様な過程、すなわち知識の抽出、知識の変換、知識の整理・体系化の各段階に対する支援ツールを、特定のタスク向けに定義したものが提案されている。

知識獲得においては、現状システムのタスク分析・整理が進められてきた経過がある。一般にエキスパート・システム構築ツールで用意されている知識表現は、プロダクション・ルールやフレームで記述され知識と推論機構の組み合わせで構成される非常に一般的な表現である。したがって対象とする問題の性質には無関係にプロダクション・システムという一つの推論方式が適用されることになる。このことは広い範囲の知識を表現することはできるが、知識の実現方法が一義には定まらないことを意味する。この問題を解決しようとする観点から知識表現を見直したのが Chandrasekaran[5] が提唱した汎化タスク（Generic task）である。汎化タスクでは各タスクに問題解決方式の安定化と問題解決知識の分析を行う。すなわち、汎化タスクはドメインに独立で、タスクに固有の基本的なタスクである。これらの結果は、特定のタスクに対象を特化したエキスパート・システム構築ツールすなわちドメインシェル[6] を導出した。これらは、知識獲得の効率化のための手法として開発されたツールであり、基本的には知識提供者との間の対話を通じて行われることからインタビュー（質問の生成）システム[7] と呼ばれている。

インタビューシステムには大きく分けて、①状況や抽出できた知識に無関係に、定型的な質問を行って知識を獲得する方法、②心理学的な方法によって、専門家に知識の連想を促す連想的知識獲得方法、③知識表現の整理や変換により、専門家自身が再確認して不足の可能性や誤謬を認識する方法、④知識獲得システムがあらかじめ獲得すべき知識がどのようなものであるかを知っていて、その知識に従ってインタビューを行う知識獲得方法などがある。④に関しては Kahn[8], McDermott[9] 等の知識獲得支援ツール MORE および MOLE がある。MORE では、8種類の戦略が用意され、専門家にインタビューを施し、診断型の知識の獲得を支援するツールである。ここではインタビューによる知識ベースの洗練化が行われると直ちにドメインモデルから診断ルールが生成されていた。しかしながら、洗練化されたと思われる知識ベースにも多くの不十分な点を含んでいた。ただ、MORE における知識表現は診断に特化されたものであるが、診断型の問題でさえあれば診断の対象には依存しない一般的な手法で知識を表現することができた。MORE でのモデル表現はルールよりは「深い知識」ということができるが問題領域そのものを理解するという観点から見れば、限りなくルールに近い「深い知識」である。エキスパートシステムに関していえることは、現状の知識ベース内の知識は特定の問題を解くことを前提とした知識である。したがって、同一ドメインの問題であっても対象に変化をきたすとその知識はまったく利用することができない。タスクに直結する知識とドメインにだけ依存する知識とが分離されていないのがその原因である[10]。「浅い知識」は専門家自身の経験と深い知識の変換から獲得できる。「浅い知識」表現は「浅い知識」を IF—THEN ルールやフレームで表現したもの、あるいは「深い知識」を特定の目的のために解釈して異なる表現法を用いたものである。また、「深い知識」表現は、「深い知識」を多様な解釈ができるような形式で表現したものである。つまり、「深い知識」は「浅い知識」を数種類の知識に分解したものであるということができ

る。共有可能性・再利用可能性が高い知識としてドメインにおける原理・原則や対象物の構造や部品に関する知識等がタスクから分離されたものとして考えられる。

エキスパートシステムのために必要な知識の2つめとして「常識」がある（上掲書10)。「常識」に関しては、常識推論や default logic などの推論に関する理論だけではない。そして、常識、ドメイン知識とタスク知識の共通点はそれらの記述を支えるオントロジーの整備である。オントロジーとは人工知能分野においては、物事を表現する際の基本となる最小の単位の集合という意味で用いている。常識の記述に必要なオントロジーとは時間、空間、因果といったような概念とそれに関連する現象や法則を記述する際に不可欠となる基本概念である[11]。

前述した MORE はさらに改良がなされて MOLE と呼ばれるシステムに発展した。MORE で行われていた知識ベースの洗練は、静的なものであったが、MOLE では実際の問題を専門家とともに解決を試み、その中で対話をするという動的な洗練を行っている。すなわち、MOLE ではインタビューのみで行う知識獲得を静的分析 (static analysis) と定義し、エキスパートシステムを試作したうえで実際の問題を専門家と共同で解きつつ行う知識獲得を動的解析 (dynamic analysis) と定義し、両者を区別した。この MOLE で示されたアプローチはソフトウェア開発の仕様獲得（要求獲得）過程で利用することができよう。ところで、仕様獲得（要求獲得）に対しては知識獲得のような定説がないのが現状であり、強いていえば、ソフトウェアシステムの導入によって解決しようとする問題記述の獲得、またはユーザ自身の独自の用語で記述したソフトウェアシステムの仕様の獲得（要求仕様獲得）であり、もう一つは要求仕様をシステムエンジニア (SE) の独自の用語で言い換えたものの獲得（設計仕様獲得）といえよう。つぎに、要求仕様獲得と知識獲得のアナロジの観点からエキスパートシステム (ES) とソフトウェアシステム開発の関連性をみた場合、SE と要求者間の対話から要求仕様獲得と知識獲得のアナロジが検出される。なぜなら、要求仕様獲得は要求者から SE への移行であり知識獲得は専門家から KE への移行であるからである。要求者は当然のことながら自らの要求に関する専門家であると考えられるから、要求仕様獲得は知識獲得の範疇にあると結論づけられる。

要求者と SE 間の対話を人間系の果たすタスクに関する知識の獲得と考えると、設計仕様獲得と知識獲得のアナロジが検出される。設計仕様獲得と知識獲得における関係は、両者とも構築すべきシステムの問題領域依存部分の獲得すべきものというアナロジにこの場合したがう必要がある。既に実用化されている 4GL[12] を利用した設計仕様獲得は知識獲得に近いものといえる。要求仕様獲得と知識獲得の関係では両者とも対話相手のタスクに関する知識を獲得するものであるというアナロジに従う必要があろう。それは、知識獲得におけるタスクごとの専門家モデルの存在からの類推をおこなうと、問題領域に依存しない要求仕様の概形を準備すればよいことになるからである。類推問題では事例の情報構造をどこまで抽象化・汎化するか出現する語彙の間の類似性をどのように評価するか等が問題となる。

2. 類　　推

類推 (Analogical Reasoning) とは、与えられた幾つかの対象間に類似性（類比）を検出し、その類比を用いて一方の対象で成立している事実や知識を、もう一方の対象に変換することにより、問題解決の手掛かりを得たり、未知の事実などを予測・推定する推論方式のことである[13]。すなわち、類似した前提が成立しているときに、類似した結論を推論することである。

予測推定型の類推としては、Winston[14] の因果関係による類推がある。この予測推定型の類推では、対象間に類比が観測されるときに類比に基づく知識の変換によって新たな事実をそれが成立するかもしれないと予測推定する。彼の類推では『類比は因果 (cause) 関係を保存する』という基本仮定にしたがっている。これを表現すると図1のようになる。この図においてa, b, a′, b′……などは対象フレーム、r_1, r_2, ……などはフレーム間の関係を示す。なお、破線はリンクである。S_1ではr_1 (a, b) がr_2 (c, d) の原因になっており、S_2ではr_1 (a, b) とリンクによって類似なr_1 (a′,

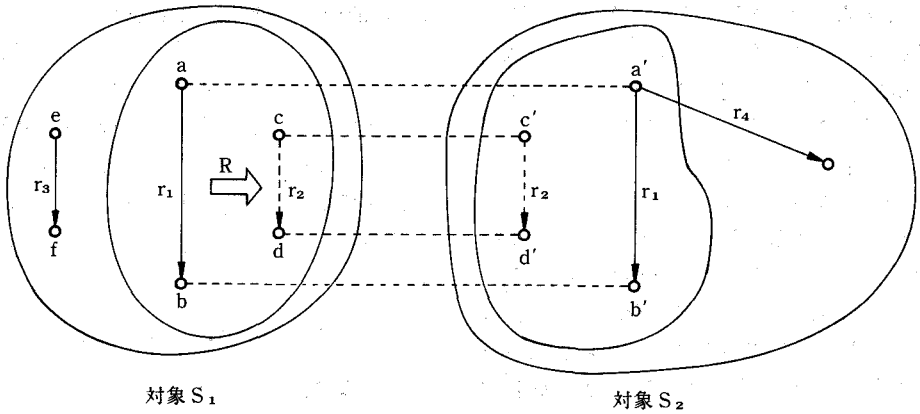

図1　Winston の類推原理

対象　S_1：前提　　α_1、……、α_n ⟶ 結論　α

類似性　ϕ

対象　S_2：前提　　β_1、……、β_n ⋯→ 結論　β？

図2　類推原理

b′）が成立している。このとき、S_1 の因果関係Rを援用して、S_2における類似な関係 r_2 (c′, d′) を推論する。

Winston は、前提と結論を因果関係における原因と結果として捉えていたが、これを論理的含意関係における条件と考え、確定節を対象にすれば、類推を述語論理の枠組みの中で定式化することができる。有川（上掲書13）によれば類推原理について図2のように明示し、類比 ϕ の定義、与えられた対象 S_1 と S_2 から ϕ を求める方法、その ϕ にしたがって図2の $\alpha\,\phi\,\beta$ なる β を推論するための操作を与える必要があると結論付けている。

一方、予測推定型の類推に対して、Carbonel[15]は問題解決型の類推を提唱し、従来の問題解決手法と類推によるそれとを統合して問題解決を行うというものである。彼は一般問題解決器（General Problem Solver）の特別な作用素（operator）として、類比に基づく「T—作用素」を導入することで、類推と従来の作用素適用が統合化できると主張した。Carbonell はまた検索された事例の修正・適合に関して、解の変形類推（transformational analogy）と誘導類推[19]（derivational analogy）を提案した。これらは、次のように記述できる。

(1)変形類推・・・事例から関係の深い解を検索し、新しい問題状況の必要条件を満足するように変換オペレータを適用し、解を徐々に変形していく。すなわち、問題解決によって得られる解そのものに有用な類似性が存在する。

(2)誘導類推・・・事例上において解の誘導過程を変換し、問題の解を再び誘導することを基本としているものであるが、これは解を得るために用いられる推論過程上、表面的な事例の特徴にはあらわれない隠れた判断プロセスの中に有用な類似性があるという考え方に基づくものである[16]。これは次のような過程をとっている。即ち、①何らかの手段で問題を解決しようとする際にはそれらの過程において採用した各段階を記憶しておく、②新しい問題に遭遇して、それにある計画を適用して解決しない場合、非類推的方法を用いて問題の解析をはじめる。③解析開始後、問題の判断過程が過去において得たそれと一致すれば、その過去の問題についてのすべての判断過程を想起して④をはじめる。もし記憶なしとなれば解の類推的変換や②の手法を試みたりする。④想起された誘導類推過程の各段階について、その判断が新たな状況のもとにおいても有効か否かを調べ有効でない部分についてはその代替案を探す。代替案もなければ③に戻る。⑤誘導過程全体が新しい問題に適合したら、それを次の機会のために記憶しておく。これらのことから誘導類推は、計画や設計などの合成型問題のモデルに適しているといえる。

変形類推の適用事例としては、Williams によ

る Teaching Assistant[17]（以下 TA と略記）がある。TA は過去に遭遇したプログラムを事例ベース（特徴付けられた問題解決事例の集まりであり、成功事例だけでなく、失敗事例も含まれる）に格納しておき、新しくプログラムを作成する際には、事例ベースから過去のプログラム事例を検索し、必要度により修正を加えつつプログラミングを学習するシステムということができよう。TA ではシステムが「プログラムを理解」し、体系化したものを格納しているプログラム事例ベースを持つことになるが、「プログラムを理解」するとは何を意味するのかを明確にしておく必要がある。この点に関しては、上原[18]の論文がある。

彼によれば次の3種類の能力が必要であるとし、同時に TA の動作概要についても論述している。

3種類の能力とは

(1)エミュレーション能力（emulation skill）

(2)説明能力（explanatory skill）

(3)類推能力（analogical skill）

であり、(1)はプログラムの仕様が与えられると作成すべきプログラムがどのような出力を生成するか予測可能なこと。(2)は予測された出力がどのように生成されたかを1ステップ毎に説明可能なこと。(3)はプログラムの仕様が与えられると仕様と類似しているプログラムの仕様を発見可能なこと

を指摘している。また、TA の動作概要については具体例を用いて説明しているが、それは次のような動作過程を採用している。

(1)類推能力を用いて与えられた仕様と類似する仕様を持つプログラムをプログラム事例ベースから検索する。

(2)エミュレーション能力を用いて、与えられた仕様から予測されるプログラムの振舞いと類似したプログラムの実際の振舞いを比較する。

(3)もし、予測された振舞いと実際の振舞いが異なっていれば、説明能力を用いてプログラムの中のどのコードが誤っているかを指摘し、さらに類推能力を用いて正確なコードを発見する。

これまでに述べてきた類推では、ある既知の知識の一部が未知の知識に写像されるという定式化を行うことであるといえる。この類推と全く同じ過程をとるものに事例に基づく推論（Casebased reasoning）がある。これまで、事例の修正には類推が関係することを述べてきた。事例の修正を行うためには、いずれにしろいくらかの領域知識が必要である。使用される領域知識が経験的知識に依存する場合、事例に基づく推論はルールベース推論に接近し、構造的知識に依存する場合は事例に基づく推論はモデルベース推論に接近するといえる。

3. まとめと課題

本稿では、まず仕様獲得と知識獲得の技術的相関について考察した。要求仕様獲得と知識獲得とは明らかに異なるフェーズである。しかし、要求仕様獲得における知識獲得の知見の適用という観点から見れば、要求仕様獲得は対話相手のタスクに関する知識の獲得という共通点がある。知識獲得の知見を要求仕様獲得に生かすには、両者とも対話相手のタスクに関する知識を獲得するものだというアナロジに従う必要がある。

次に、ユーザ要求仕様が完全で、逆に設計知識が不完全な問題では、既存の設計知識と欠落した知識の間に類比を探し出し、その類比から既存の知識を欠落した知識に適合するように修正する類推による問題解決が利用可能なものと思われる。知識獲得は知識ベース構築方法論と深い関連があり、またエキスパートが実際に扱っている問題の構造を反映した汎化タスクを組み合わせることも考慮にいれながら、構築方法論の今後の問題解決の理論的手がかりとしたい。

（たにぐち みちおき 教授）

（1996. 1. 8 受理）

参考文献

1） Marca, D. A. and McGowan, C. L. : SADT ™: *Structured Analysis and Design Technique*, McGraw-Hill, Inc., 1988.

2） 鈴木健蔵他：「SE の仕様獲得の実際」『情報処理』, Vol.33, No.6, pp.612–616, 1992.

3） Otsuki, S. and Takeuchi, A.: Intelligent CAL system Based on Teaching Strategy and Learner Model, *proc. IFIP WCCE'85*, North-Holland pp.463–468.

4） 上原邦昭：「要求獲得の知的支援」『人工知能学会誌』, Vol.6, No.2, p.171, 1991.

5） Chandrasekaran, B.: Generic task in knowl-

edgebased reasoning: High-level building blocks for expert system design, *IEEE Expert*, pp.23–30, Fall, 1986.

6) 人工知能学会（編）:『人工知能ハンドブック』chapter7-1, pp.570–592, オーム社 (1990).

7) 溝口理一郎、角所収:「知識獲得支援システム」(『人工知能学会誌』、Vol.3, No.6, pp.732–740, 1988).

8) Kahn, G., Nowlan, S. and McDermott, J. : Strategies for knowledge acquisition, *IEEE Trans. Pattern Analysis and Machine Intell.*, vol. PAMI-7, No.5, pp.511–522, 1985.

9) Eshelman, L. and McDermott, J.: A knowledge acquisition tool that uses its head, *Proc. of AAAI'86*, pp.950–955, 1986.

10) ティヘリノ・ジュリ・A、他:「タスクオントロジーと知識再利用に基づくエキスパートシステム構築方法論—タスク解析インタビューシステム MU-LTIS の基本構想—」(『人工知能学会誌』 Vol. 8, No.4, pp.476–487, 1993).

11) 溝口理一郎:「知識の共有と再利用研究の現状と動向」(『人工知能学会誌』 Vol.9, No.1, pp.3–9, 1994).

12) Sommerville. I.: *Requirement Validation and Prototyping*, chapter 6, pp.109–122, Addison Wesley, Third edition, 1989.

13) 有川節夫:「知識情報処理の基本技術 3-1 述語論理と推論」(『信学誌』、Vol.69, No.11, p.1117).

14) Winston, P. H.: Learning New Principles from Precedents and Excercises, *Artif. Intel. 19*, pp.321–350, 1983.

15) Carbonell, J. G. : A computational Model of Analogical Problem Solving, *IJCAI-81*, pp.147–152, 1981.

16) Carbonell, J. G. : Learning by Analogy: Formulating and Generalizing Plan from Past Experience, in Michalski, R. S., Carbonell, J. G. and Mitchell, T. M., *Machine Learning, An Artificial Intelligence Approach*, Vol.1, pp.371–392, Morgan Kaufmann, 1983.

17) Williams, R. S. : Learning to Program by Examining and Modifying Cases, *Proc. of the 5th International Conference on Machine Learning*, pp.318–324, 1988.

18) 上原邦昭:「ソフトウェア設計における仕様獲得支援」(『情報処理』 Vol.33, No.6, p.633, 1992).

19) Carbonell J. G. : Derivational analogy : a theory of reconstructive problem solving and expertise acquisition, in Michalski, R. S., *et al.* eds. *Machine learning: an artificial intelligence approach* II, Morgan Kaufmann, pp.371–392, 1986. 邦訳:電総研 AI 研究グループ訳:『誘導類推:再生型問題解決と専門知識獲得に関する理論、in 類推学習』、共立出版、1988